# GPUMCD: a new GPU-Oriented Monte Carlo dose calculation platform

Sami Hissoiny* and Benoît Ozell

*École polytechnique de Montréal, Département de génie informatique et génie logiciel, 2500 chemin de Polytechnique. Montréal (Québec), CANADA H3T 1J4*

Hugo Bouchard and Philippe Després

*Département de Physique, Université de Montréal, Pavillon Roger-Gaudry (D-428), 2900 Boulevard Édouard-Montpetit, Montréal, Québec H3T 1J4, Canada and Département de Radio-oncologie, Centre hospitalier de l'Université de Montréal (CHUM), 1560 Sherbrooke est, Montréal, Québec H2L 4M1, Canada*

(Dated: January 7, 2011)

# Abstract

**Purpose :** Monte Carlo methods are considered the gold standard for dosimetric computations in radiotherapy. Their execution time is however still an obstacle to the routine use of Monte Carlo packages in a clinical setting. To address this problem, a completely new, and designed from the ground up for the GPU, Monte Carlo dose calculation package for voxelized geometries is proposed: GPUMCD.

**Method :** GPUMCD implements a coupled photon-electron Monte Carlo simulation for energies in the range 0.01 MeV to 20 MeV. An analogue simulation of photon interactions is used and a Class II condensed history method has been implemented for the simulation of electrons. A new GPU random number generator, some divergence reduction methods as well as other optimization strategies are also described. GPUMCD was run on a NVIDIA GTX480 while single threaded implementations of EGSnrc and DPM were run on an Intel Core i7 860.

**Results :** Dosimetric results obtained with GPUMCD were compared to EGSnrc. In all but one test case, 98% or more of all significant voxels passed a gamma criteria of 2%-2mm. In terms of execution speed and efficiency, GPUMCD is more than 900 times faster than EGSnrc and more than 200 times faster than DPM, a Monte Carlo package aiming fast executions. Absolute execution times of less than 0.3 s are found for the simulation of 1M electrons and 4M photons in water for monoenergetic beams of 15 MeV, including GPU-CPU memory transfers.

**Conclusion :** GPUMCD, a new GPU-oriented Monte Carlo dose calculation platform, has been compared to EGSnrc and DPM in terms of dosimetric results and execution speed. Its accuracy and speed make it an interesting solution for full Monte Carlo dose calculation in radiation oncology.



Keywords: Monte Carlo, GPU, CUDA, EGSnrc, DPM, DOSXYZnrc,



* sami.hissoiny@polymtl.ca

## I. INTRODUCTION

Monte Carlo packages such as EGS4 [1], EGSnrc [2, 3], EGS5 [4] and PENELOPE [5, 6] have been extensively compared to experimental data in a large array of conditions, and generally demonstrate excellent agreement with measurements. EGSnrc, for instance, was shown to "pass the Fano cavity test at the 0.1% level" [7]. Despite this accuracy, Monte Carlo platforms are mostly absent from routine dose calculation in radiation therapy due to the long computation time required to achieve sufficient statistical significance.

Monte Carlo simulations are generally considered the gold standard for benchmarking analytical dose calculation approaches such as pencil-beam based computations [8–10] and convolution-superposition techniques [11, 12]. Such techniques typically use precomputed Monte Carlo data to incorporate physics-rich elements in the dose calculation process. These semi-empirical methods were developed as a trade-off between accuracy and computation time and as such do not match the level of accuracy offered by Monte Carlo simulations, especially in complex heterogeneous geometries where differences of up to 10% were reported [13–15].

Fast Monte Carlo platforms have been developed for the specific purpose of radiotherapy dose calculations, namely VMC [16–18] and DPM [19]. These packages make some sacrifices to the generality and absolute accuracy of the simulation by, for example, restricting the energy range of incoming particles. This assumption allows simpler treatment of certain particle interactions which are less relevant within the limited energy range considered. Gains in efficiency of around 50 times can be achieved by these packages when compared to general-purpose solutions.

This paper presents GPUMCD (GPU Monte Carlo Dose), a new code that follows the fast Monte Carlo approach. GPUMCD relies on a relatively new hardware platform for computations: the GPU (Graphics Processing Unit). The GPU is gaining momentum in medical physics [20–23] as well as in other spheres [24–28] where high-performance computing is required. A layer-oriented simulation based on the MCML Monte Carlo code for photon transport has already been ported to the GPU [29]. The photon transport algorithm from PENELOPE has also been ported to the GPU [30] for accelerations of up to 27x. DPM [19] has been ported to the GPU with excellent dosimetric results compared to the CPU version, as expected, but with relatively low (5-6.6x) acceleration factors [31].

A comparison of GPUMCD to the work by Jia *et al.* must be made cautiously. The work of Jia *et al.* is a port of an existing algorithm that is not, to the best of our knowledge, designed for the specific nature of the GPU. Notable differences are found between the two approaches regarding the treatment of secondary particles and the electron-photon coupling. These differences will be exposed in Sec. II C 3. The code presented in this article is the first attempt of a complete rewrite of a coupled photon-electron Monte Carlo code specifically designed for the GPU. New techniques for the memory management of particles as well as efforts to reduce the inherent divergent nature of Monte Carlo algorithms are detailed. Divergence in GPU programming arises from conditional branching, which is not optimal in stream processing where predictable execution is expected for optimal performances. No new sampling methods used in Monte Carlo simulations were introduced in GPUMCD; all theoretical developments of cross section sampling methods have been adapted from their description in general-purpose Monte Carlo package manuals (and references therein) listed previously. However, their actual implementation is original, as it is specifically tailored for parallel execution on graphics hardware, and the code has not been taken from existing Monte Carlo implementations.

The paper is structured as follows: Sec. II introduces the physics principles as well as the hardware platform and implementation details. Sec. III presents the results in terms of dose and calculation efficiency of GPUMCD compared to the EGSnrc and DPM platforms. Finally, a discussion of the reported results is presented in Sec. IV.

## II. MATERIAL AND METHODS

### A. Photons simulation

Photon transport can be modeled with an analogue simulation, *i.e.* every interaction is modeled independently until the particle leaves the geometry or its energy falls below a certain energy level referenced as $P_{cut}$ from now on. This allows for an easy implementation of the transport process.

GPUMCD takes into account Compton scattering, the photoelectric effect and pair production. Rayleigh scattering can be safely ignored for the energy range considered (10 keV-20 MeV). Considering a homogeneous phantom, the distance to the next interaction, noted

$s$, can be sampled from the following expression:

$$s = -\frac{1}{\mu(E)}\ln(\zeta), \tag{1}$$

where $\zeta$ is a random variable uniformly distributed in (0,1] and $\mu(E)$ is the value of the total attenuation coefficient, given by

$$\mu = \frac{N_A}{A}\rho\left(\sigma_{compton} + \sigma_{photo} + \sigma_{pair}\right), \tag{2}$$

where the $\sigma$'s are total cross section values for the corresponding interactions, $N_A$ is Avogadro's number, $A$ is the molecular weight and $\rho$ is the density.

To eliminate the need for a distinct geometry engine in heterogeneous situations, GPUMCD employs the Woodcock raytracing algorithm [32] in which the volume can be considered homogeneously attenuating by introducing a fictitious attenuation coefficient, corresponding to a fictitious interaction which leaves the direction and energy of the particle unchanged, in every voxel $\vec{x}$:

$$\mu(\vec{x})_{fict} = \mu_{max} - \mu(\vec{x}) \tag{3}$$

where $\mu_{max}$ is the attenuation coefficient of the most attenuating voxel inside the volume. The distance to the next interaction is therefore always sampled using $\mu_{max}$. The sampling of the interaction type, once at the interaction position, is then performed taking into consideration this fictitious interaction type. This method is not an approximation, it leads to the correct result even in arbitrarily heterogeneous geometries.

Compton scattering is modeled with a free electron approximation using the Klein-Nishina cross section [33]. The energy and direction of the scattered photon and secondary electron are sampled according to the direct method derived by Everett *et al.* [34]. Binding effects and atomic relaxation are not modeled. These approximations are reasonable when the particle energy is large compared to the electron binding energy which is the case for radiation therapy [35].

The photoelectric effect is modeled by once again ignoring atomic relaxation. Additionally, shell sampling is also ignored and all electrons are assumed to be ejected from the K-shell. The sole product of the interaction is a new electron in motion with total energy $E_{elec} = E_{phot}$. The angular sampling of the electron is selected according to the Sauter distribution [36] following the sampling formula presented in Sec. 2.2 of the PENELOPE

manual [5]. The simulation of the photoelectric effect will become less accurate when the energy of the photon is in the order of the characteristic X-ray radiation energy of the material since characteristic X-rays are not produced. Therefore, at the previously stated lower energy range threshold of 10 keV, photons are still tracked and simulated but the accuracy of the simulation of the photoelectric effect will be dependent on the material where the simulation is taking place. The binding energies being low in organic compound, this approximation is considered acceptable.

Relatively crude approximations are employed when simulating pair production events. First and foremost, no positrons are simulated in this package. The pair production event instead generates two secondary electrons. The relatively low probability of pair production events at the energies considered as well as the additional low probability of in-flight annihilation of the positron make this approximation suitable [19]. Triplet production is also not modeled. The energy of the incoming photon is trivially split between the two electrons using a uniformly distributed random number. The angle of both electrons is sampled using the algorithm presented in Eq. 2.1.18 of the EGSnrc manual [37].

## B. Electron simulation

Because of the much larger number of interactions experienced by a charged particle before it has deposited all of its energy, analogue simulations cannot be used practically. A class II condensed history method is used here, as defined by Berger [38], in which interactions are simulated explicitly only if they cause a change in orientation greater than $\theta_c$ or a change in energy larger than $E_c$. Below these thresholds, the interactions are modeled as taking part of a larger condensed interaction, the result of which is a major change in direction and energy. Above the thresholds, interactions, usually called hard or catastrophic interactions, are modeled in an analogue manner similar to the simulation of photons.

### *1. Hard interactions*

GPUMCD simulates inelastic collisions as well as bremsstrahlung in an analogue manner, as long as the changes to the state of the particle are greater than the previously discussed threshold.

For inelastic collisions, a free electron approximation is used and no electron impact ionization nor spin effects are modeled. The Møller cross section is used to describe the change in energy and direction of the scattered and knock-on electrons. The sampling routine presented in Sec. 2.4.3.i of the EGSnrc manual [37] is used.

During a bremsstrahlung event, the energy of the created photon is sampled as presented in Sec. 2.4.2.ii of the EGSnrc manual. Approximations to the angular events are employed : the angle of the electron is unchanged and the angle of the photon is selected as $\theta = m_0c^2/E$ where $E$ is the energy of the incoming electron.

### *2. Multiple scattering*

The class II condensed history method requires the selection of an angle for the multiply-scattered electron after a step, as discussed in the next subsection. To this end, the formulation presented by Kawrakow and Bielajew [39] and Kawrakow [2] is used. The required $q_{SR}^{2+}$ data are imported from the `msnew.dat` file packaged with EGSnrc.

### *3. Electrons transport*

The transport of electrons is less straightforward than photon transport because of the high probability of soft collisions, which are regrouped in a single step between two hard interactions (collision or bremsstrahlung events). This condensed history method introduces an unphysical (but mathematically converging) factor for the length of the multiple-scattering step. Between consecutive hard collisions, which are separated by a distance sampled with an electron version of Eq. (1), a number of multiple-scattering events will occur. In GPUMCD, a simple step-length selection (henceforth $e_s$) is used and can be summarized as

$$e_s = min\{d_{vox}, e_{s-max}, e_{hard}\} \tag{4}$$

where $d_{vox}$ is the distance to the next voxel boundary, $e_{s-max}$ is a user-supplied upper bound of step length and $e_{hard}$ is the distance to the next hard interaction.

The distance to the next hard interaction can once again be sampled with the help of the Woodcock raytracing algorithm. For electron transport, however, the total attenuation coefficient, $\mu(E)$, is in fact $\mu(E, s)$ where the dependence on distance has been made explicit.

The dependence on distance is due to the fact that electrons will lose energy due to sub-threshold interactions between two successive hard interactions. In this application, the approximation is made that the attenuation coefficient is decreasing with respect to the particle energy. It is not rigorously true, as shown in [2], but allows for an easier treatment since $\mu_{max}(E)$ can be selected with $E$ being the energy at the beginning of the electron step.

Energy losses are experienced by the electron during the soft scattering events that are modeled by the multiple-scattering step. This energy loss is accounted for with the continuously slowing down approximation (CSDA). This approximation states that charged particles are continuously being slowed down due to the interactions they go through and that the rate of energy loss is equal to the stopping power $S$, given by:

$$S(E) = -\frac{dE}{ds}. \tag{5}$$

The implementation of a class II condensed history method only accounts for sub-threshold interactions in the CSDA. The use of restricted stopping powers, $L$, is therefore required:

$$L(E, K_{cut}) = \int_0^{K_{cut}} \Sigma(E, E')dE' \tag{6}$$

where $\Sigma(E, E')$ is the total macroscopic cross section and $K_{cut}$ is the catastrophic interaction kinetic energy threshold. Since this release of GPUMCD is $e_s$-centric, the energy loss of a given step is computed with

$$\Delta E = \int_0^{e_s} L(s)ds. \tag{7}$$

The evaluation of the integral can be reduced to

$$\Delta E = L\left(E_0 - L(E_0)s/2\right) \tag{8}$$

which is EGSnrc's Eq. 4.11.3.

Between two consecutive hard collisions, the electron does not follow a straight line. Bielajew's alternate random hinge method [5] builds on PENELOPE's random hinge method to handle the angular deflection and lateral displacement experienced by an electron during an electron step. The random hinge method can be described as follows: the electron is first transported by $\zeta e_s$ (where $\zeta$ is again a uniformly distributed random number), at which point the electron is rotated by the sampled multiple- scattering angle described in Sec. II B 2 and the energy loss is deposited. The electron is then transported for the remaining distance equal to $(\zeta - 1)e_s$. Bielajew's refinement of the algorithm involves randomly sampling the

angular deflection either before or after the electron has deposited the energy associated with the step.

### C. GPU implementation

In this section, different aspects of the GPU implementation are detailed. GPUMCD is built with the CUDA framework from NVIDIA. A general description of the GPU building blocks and memory levels can be found in [40]. Sec. II C 1 presents the random number generator used in this work while Sec. II C 2 describes the memory management. Finally, Sec. II C 3 addresses the SIMD divergence issue and presents solutions as well as other optimizations used in GPUMCD.

#### *1. Random number generator*

A pseudo random number generator (PRNG) is already available in the NVIDIA SDK (Software Development Kit) for GPU computing. However, it uses a lot of resources which would then be unavailable for the rest of the Monte Carlo simulation. For this reason, a new lightweight PRNG based on the work of Marsaglia [41] was implemented. We use a combined multiply-with-carry (MWC) generator, with recurrence of the form

$$x_{n+1} = (a * x_n + c_n) \bmod (b) \tag{9}$$

where $a$ is the multiplier and $b$ the base. The carry, $c$, is defined by:

$$c_{n+1} = \lfloor \frac{(a * x_n + c_n)}{b} \rfloor. \tag{10}$$

The carry $c$ is naturally computed with integer arithmetic and bases $b$ of $2^{32}$ or $2^{16}$. The choice of the multiplier $a$ is not arbitrary: the multiplier is chosen so that $ab - 1$ is safe prime. The period of the PRNG with such a multiplier is on the order of $(ab - 1)/2$. This PRNG has the advantage of using a small amount of registers. By combining two of these generators for each thread, the PRNG uses two 32 bits values for the multipliers and two 32 bits values for the current state. Ten integer operations (3 $shifts$, 2 $ands$, 2 $mults$ and 3 $adds$) are required to generate one new number. It has been shown to pass all tests but one of the TestU01 suite [42], a program designed to test the quality of pseudo random numbers, and to achieve 96.7% of the peak integer bandwidth.

*2. Memory management*

GPUMCD is composed of four main arrays: a list of electrons and a list of photons, an array to store composition identifiers and another for density values corresponding to a numerical phantom. During the initialization phase, a call to the `initParticle` function is made which fills the correct particle array (depending on the source particle type) and according to the source type (*e.g.* with a parallel beam source all particles have a direction vector of (0,-1,0)). The particle arrays are allocated only once, with size `MAXSIZE`. The choice of `MAXSIZE` is graphics card dependent as the particle arrays occupy the better part of the global memory, the largest pool of memory on the graphics card. For instance, on a 1 Gb card, `MAXSIZE` could be set to $2^{20}$. Only a fraction of the particle array will be filled with source particles, to leave room for secondary particle creation. If for instance `MAXSIZE`$/d$ particles are generated, then each primary particle can generate an average of $d$ secondary particle without running out of memory. The choice of $d$ is therefore important; it is energy dependent and to a lesser extent geometry dependent. For example, with a monoenergetic 15 MeV beam of photons on a 32 cm deep geometry, a value of $d = 8$ was found to be sufficient to accommodate all secondary particles. A global counter of active particles is kept for both electrons and photons and every time a new particle is added the counter is atomically incremented, until it is equal to `MAXSIZE` after which secondary particles are discarded. If the choice of $d$ and `MAXSIZE` are such that the number of particles generated is lower than `MAXSIZE`, then no particles are discarded. Atomic operations are a mechanism that ensure that two parallel threads cannot write the same memory location at the same time, therefore discarding one thread's modification. Atomic operations are not ideal on parallel hardware but these two integer values, the current number of particles of both types, should be efficiently cached on GF100 class hardware through the use of the GPU-wide 768 Kb of L2 cache. After the initialization phase is completed, the simulation starts by calling the function to simulate the initial particle type. Subsequently, secondary particles of the other type are generated and put inside the particle array for that type. The other type of particle will then be simulated, and so on until a relatively low number of particle is left in the stack. This number is user-selectable and could be set to 0 to simulate every particle. Since a simulation pass always comes with some overhead, it could also be set to a non-zero value to avoid the launching of a simulation pass with few particles to

simulate which could have a negligible impact on the final distribution. If this impact is in fact negligible of course depends on the threshold selected. Photons and electrons are never simulated in parallel but instead the particles of the *other* type are placed in their respective array and wait there until the array with the *current* type of particle is exhausted. This, in turn, eliminates the divergence due to the photon-electron coupling. Every time a call to a simulation function for electrons or photons is issued, one can consider that every particle of that type is simulated and that the counter for that type of particle is reset to 0. Sec. II C 3 will show some refinements related to the management of particles.

The other two main arrays, one of composition identifier and one of density values, are stored as 3D textures since they are used to represent the volume of voxels. 3D textures reside in memory but they can be efficiently cached for 3D data locality. Every time a particle is moved, the composition and density of the current particle voxel are fetched from the 3D textures.

Cross section and stopping power data are stored in global memory in the form of a 1D texture. The cross section and stopping power data are preinterpolated on a regular grid with a value at every 1 keV.

The shared memory usage is relatively limited in this application. For the electrons simulation, some specific composition attributes are required (*e.g.* $Z_V$, $Z_G$, etc., in Tab. 2.1 of EGS5) and are stored in shared memory. These data values have not been placed in constant memory since that type of memory is best used when all threads of a warp access the same address, which cannot be guaranteed here since these composition attributes are material dependent. Two particles in different materials would therefore request the constant memory at two different addresses, resulting in serialization. No other use for shared memory has been found since the application is by nature stochastic. For instance, we cannot store a portion of the volume since there is no way to know where all the particles of one thread block will be. Similarly, we cannot store a portion of the cross-section data since there is no way to know in which material and with which energy all the particles of one thread block will be.

### 3. Divergence reduction and other optimization strategies

Every multiprocessor (MP) inside the GPU is in fact a SIMD processor and the SIMD coherency has to be kept at the warp level which is the smallest unit of parallelism and is composed of 32 threads. The Monte Carlo simulation being inherently stochastic, it is impossible to predict which path a given particle will take and it is therefore impossible to regroup particle with the same fate into the same warps. When a warp is divergent, it is split into as many subwarps as there are execution paths, leading to a performance penalty. Divergence can be seen when, *e.g.* two particles of the same warp do not require the same number of interactions before exhaustion or do not interact in the same way. Some software mechanisms can be employed to reduce the impact of divergence.

The first mechanism consists in performing a stream compaction after $N$ simulation steps, where $N$ is user-defined and corresponds to a number of interactions for photons and to a number of catastrophic events for electrons. For example, if one particle requires 20 interactions to complete and the others in the warp require less than 5, then some scalar processors (SP) in the MP will be idle while they wait for the *slow* particle to finish. This first mechanism then artificially limits the number of simulation steps a particle can undergo during one pass of the algorithm. After this number of steps, every particle that has been completely simulated is removed from the list of particles and the simulation is restarted for another $N$ steps, until every particle has been completely simulated. The removal of particles is not free and therefore it is not clear if this technique will have a positive effect on execution time. The stream compaction is accomplished with the CHAG library [43].

A second mechanism, named persistent thread by its creators and *pool* in GPUMCD, is taken from the world of graphics computing [44]. In this approach, the minimum number of threads to saturate the GPU is launched. These threads then select their workload from a global queue of particles to be simulated. Once a thread is done with one particle, it selects the next particle, until all have been simulated. This is once again to reduce the impact of the imbalance in the number of interactions per particle.

All the GPU code uses single precision floating point numbers as they offer a significant speed improvement over double precision floating point numbers on graphics hardware. The work of Jia et al. [31] showed that no floating point arithmetic artifacts were introduced for this type of application.

A notable difference between this work and the work by Jia *et al.* is the way the secondary particles are treated. In the work by Jia, a thread is responsible for its primary particle as well as every secondary particles it creates. This can be a major source of divergence because of the varying number of secondary particles created. On the other hand, GPUMCD does not immediately simulate secondary particles but instead places them in their respective particle arrays. After a given pass of the simulation is over, the arrays are checked for newly created secondary particles and if secondary particles are found, they are simulated. This eliminates the divergence due to the different number of secondary particles per primary particles. Additionally, since Jia *et al.* use a single thread per primary particles, a thread may be responsible for the simulation of both electrons and photons which can in turn be a major source of divergence. For example, at time $t$, thread $A$ may be simulating a secondary electron while thread $B$ a secondary photon. In other words, they simulate both photons and electrons at the same time. Since both of these particles most likely have completely different code path, heavy serialization will occur. On the other hand, since GPUMCD stores secondary particles in arrays to be simulated later, no such divergence due to the electron-photon coupling occurs.

Finally, GPUMCD can be configured to take advantage of a multi-GPU system. The multi-GPU approach is trivial for Monte Carlo simulations: both GPUs execute the `initParticle` function and simulate their own set of particles. After the simulations, the two resulting dose arrays are summed. Linear performance gains are expected for simulations when there are enough particles to simulate to overcome the overhead penalty resulting from copying input data to two graphics card instead of one.

These optimization mechanisms can be turned on or off at the source code level in GPUMCD.

### D. Performance evaluation

The performance evaluation of GPUMCD is twofold: a dosimetric evaluation against EGSnrc/DOSXYZnrc and an efficiency evaluation against EGSnrc and DPM, the later being a fairer comparison since DPM was designed for the same reason GPUMCD was developed, *i.e.* fast MC dose calculations.

All settings for DPM were set to default, notably $K_{cut} = 200$ keV and $P_{cut} = 50$ keV.

Most parameters for DOSXYZnrz (unless otherwise noted) were also set to default, notably ESTEPE=0.25, $\xi_{max}$=0.5. In EGSnrc/DOSXYZnrc, the boundary crossing algorithm used was PRESTA-I and the electron step algorithm was PRESTA-II, atomic relaxation was turned off as well as bound Compton scattering. Values of $P_{cut}$ and $E_{cut}$ were set to 0.01 MeV and 0.7 MeV respectively. The same $64^3$ grid with 0.5 $cm^3$ voxels is used for all simulations on all platforms. These values of spatial resolution and number of voxels are insufficient for a clinical calculation with patient specific data; however, they were judged adequate for the benchmarking study conducted here. Preliminary results suggest that no loss of efficiency occurs between GPUMCD and other platforms as the number of voxels is increased.

The uncertainty of Monte Carlo simulation results varies as $1/\sqrt{N}$ where $N$ is the number of histories. All simulations ran for visualization purposes on GPUMCD were conducted with enough primary particles to achieve a statistical uncertainty of 1% or less. The same number of particles were then generated in DOSXYZnrc. The graphs are shown without error bars since they would simply add clutter to the results. For execution time comparisons, 4 million particles were generated and simulated for photon beams and 1 million particles for electron beams. All sources were modeled as a monoenergetic parallel beam. All dose distributions were normalized with respect to $D_{max}$. The efficiency measure, $\epsilon$, was used to evaluate the performance of the different implementations:

$$\epsilon = \frac{1}{s^2 T}, \tag{11}$$

where $s$ is the statistical uncertainty and $T$ the computation time. Only voxels with a dose higher than $0.2 \cdot D_{max}$ were considered for the statistical uncertainty, yielding the following expression for the uncertainty [45]:

$$s = \frac{1}{N_{0.2}} \sum_{D_{ijk} > 0.2 D_{max}} \frac{\Delta D_{ijk}}{D_{ijk}}, \tag{12}$$

where

$$\Delta D_{ijk}^2 = \frac{\langle D_{ijk}^2 \rangle - \langle D_{ijk} \rangle^2}{N - 1}, \tag{13}$$

for $N$ the number of histories simulated, $\langle D \rangle$ the mean of the random variable $D$ and $N_{0.2}$ the number of voxel with dose higher than $0.2 D_{max}$.

For the dosimetric evaluation, a gamma index was used to compare the two dose distributions [46]. The gamma index for one voxel $x$ is defined as

$$\gamma(x) = min\{\Gamma(x, x')\}\forall\{x'\}, \tag{14}$$

and

$$\Gamma(x, x') = \sqrt{\frac{||x' - x||^2}{\Delta d^2} + \frac{|D(x) - D(x')|^2}{\Delta D^2}}, \tag{15}$$

where $||x' - x||$ is the distance between voxels $x$ and $x'$, $D(x)$ is the dose value of voxel $x$, $\Delta d$ is the distance tolerance value and $\Delta D$ is the dose tolerance value. The criteria for acceptable calculation is defined for a gamma index value below or equal to 1.0.

All simulations were run on the same PC comprising an Intel Core i7 860 and a NVIDIA GeForce GTX480 graphics card. DPM and DOSXYZnrc do not natively support multi-core architectures and have not been modified. GPUMCD, unless running in multi-GPU configuration, uses only one processor core.

## III. RESULTS

Several slab-geometry phantoms are described in Sec. III A . In Sec. III B, the execution times and gains in efficiency are reported. A multi-GPU implementation is also tested and corresponding acceleration factors are presented.

### A. Dosimetric results

For slab geometries, central axis percent depth dose (PDD) curves as well as dose profiles at different depths or isodose curves are presented. Gamma indices are calculated in the entire volume of voxels and the maximum and average values are reported. The number of voxels with a gamma value higher than 1.0 and 1.2 are also detailed in order to evaluate discrepancies in dose calculation. The gamma criteria used is set to 2% and 2 mm which is a generally accepted criteria for clinical dose calculation [47]. In every gamma index summary table: 1) $\gamma^{max}$ is the maximum gamma value for the entire volume of voxel; 2) $\gamma_c^{avg}$ is the average gamma value for voxels with $D > cD_{max}$; 3) $\Sigma D_c$ is the number of voxels with $D > cD_{max}$ ; 4) $\Sigma\gamma_c > g$ is the number (proportion) of voxels with $D > cD_{max}$ and $\gamma > g$.

The first slab geometry is a simple water phantom. GPUMCD treats homogeneous phantoms as heterogeneous phantoms; the particles are transported as if evolving inside a heterogeneous environment and therefore there is no gain in efficiency resulting from homogeneous media. For this geometry, the results for an electron beam and a photon beam are presented in Fig. 1 and gamma results in Tab. I.

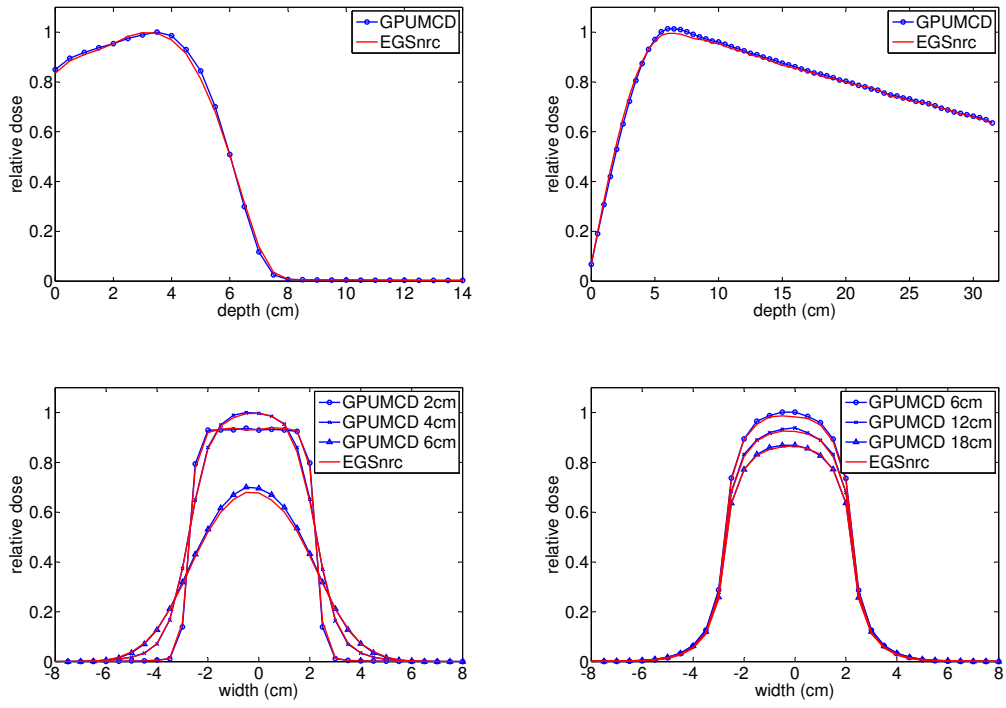


FIG. 1. PDD (top) and profile (bottom) of 15 MeV electron (left) and photon (right) beams on water.

| Particles | $\gamma^{max}$ | $\gamma^{avg}_{0.2}$ | $\Sigma D_{0.2}$ | $\Sigma\gamma_{0.2} > 1.0$ | $\Sigma\gamma_{0.2} > 1.2$ |
|---|---|---|---|---|---|
| Electrons | 1.24 | 0.29 | 1750 | 37 (2%) | 2 (∼0%) |
| Photons | 1.27 | 0.18 | 7500 | 143 (2%) | 19 (∼0%) |

TABLE I. Gamma criteria summary for a 15 MeV beam on water.

The second geometry is composed of a lung box inside a volume of water. It is designed to demonstrate the performance of GPUMCD with lateral and longitudinal disequilibrium

conditions. For electrons, the lung box is 4 cm long, 4.5 cm wide and starts at a depth of 3 cm; for photons the lung box is 6.5 cm long, 4.5 cm wide and starts at a depth of 5.5 cm. In both cases, the box is centered on the central axis. For this geometry, PDD and isodose plots are presented in Fig. 2 for the electron and photon beams while the gamma results are presented in Tab. II.

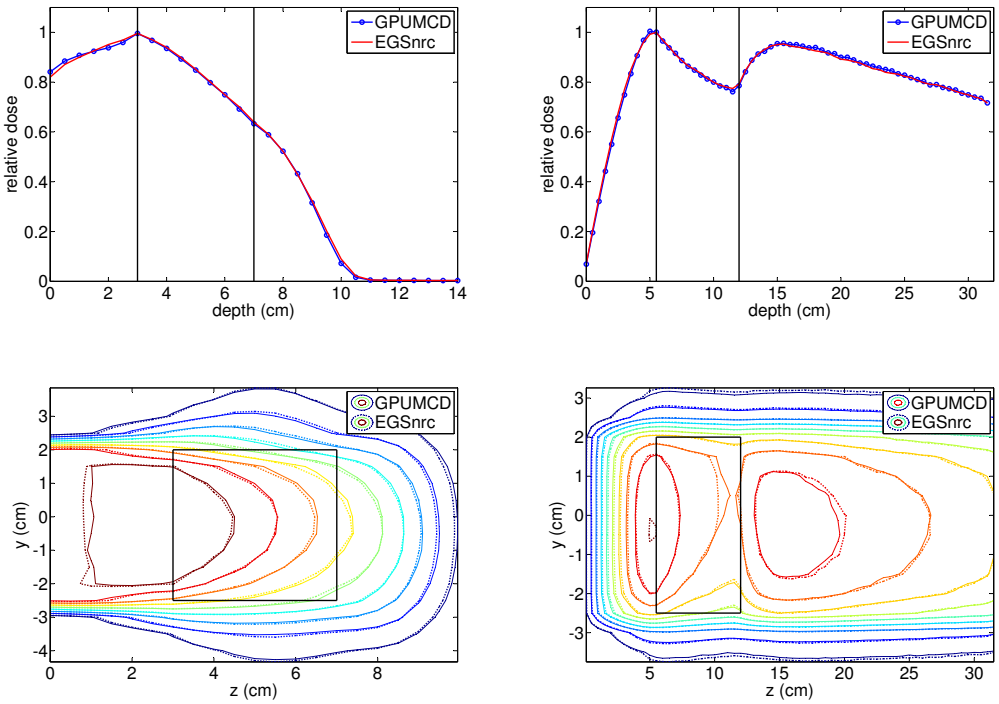


FIG. 2. PDD (top) and isodose (bottom) of a (left) 15 MeV electron beam on water with a 4 cm long, 4.5 cm wide box of lung at a depth of 3 cm and (right) 15 MeV photon beam on water with a 6.5 cm long, 4.5 cm wide box of lung at a depth of 5.5 cm.

| Particles | $\gamma^{max}$ | $\gamma^{avg}_{0.2}$ | $\Sigma D_{0.2}$ | $\Sigma\gamma_{0.2} > 1.0$ | $\Sigma\gamma_{0.2} > 1.2$ |
|---|---|---|---|---|---|
| Electrons | 1.40 | 0.32 | 2366 | 46 (2%) | 2 (0%) |
| Photons | 1.30 | 0.19 | 7522 | 122 (2%) | 19 (∼0%) |

TABLE II. Gamma criteria summary for a 15 MeV beam on water with a lung box.

The third geometry is composed of soft tissue, bone and lung. The slabs, for electrons, are arranged as such: 2 cm of soft tissue, 2 cm of lung, 2 cm of bone followed by soft tissue.

For photons the geometry is: 3 cm of soft tissue, 2 cm of lung, 7 cm of bone followed by soft tissue. The results for the electron and photon beams are presented in Fig. 3 and gamma results in Tab. III.

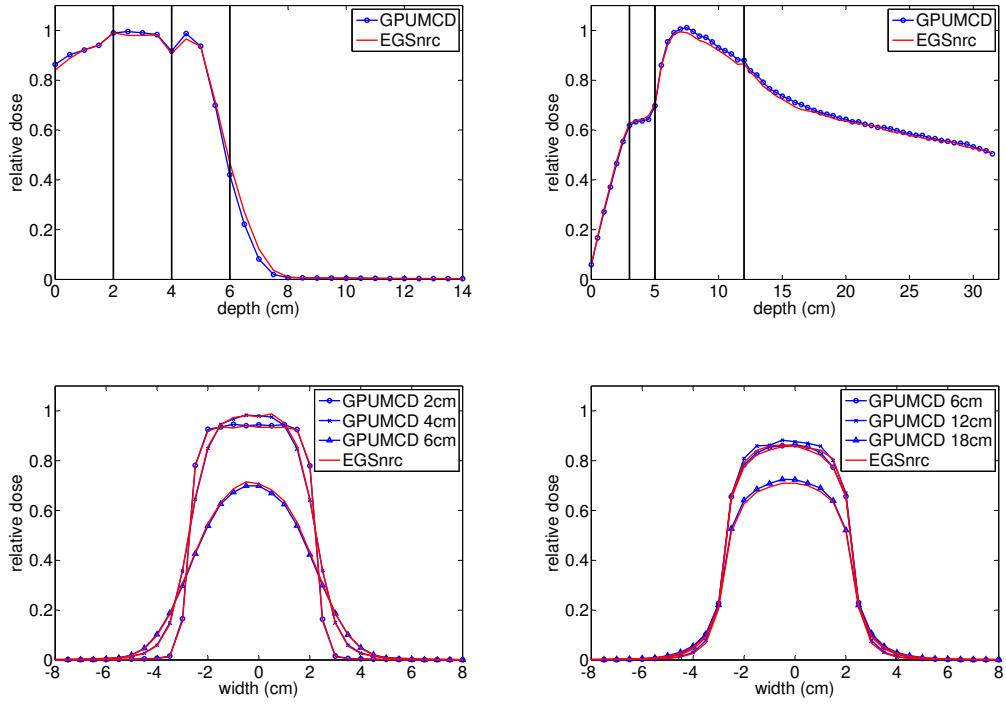


FIG. 3. PDD (top) and profile (bottom) of a (left) 15 MeV electron beam on a phantom composed of 2 cm of soft tissue, 2 cm of bone, 2 cm of lung followed by soft tissue and (right) 15 MeV photon beam on a phantom composed of 3 cm of soft tissue, 2 cm of lung, 7 cm of bone followed by soft tissue.

| Particles | $\gamma^{max}$ | $\gamma_{0.2}^{avg}$ | $\Sigma D_{0.2}$ | $\Sigma\gamma_{0.2} > 1.0$ | $\Sigma\gamma_{0.2} > 1.2$ |
|---|---|---|---|---|---|
| Electrons | 1.84 | 0.34 | 1700 | 138 (8%) | 54 (3%) |
| Photons | 1.31 | 0.22 | 7220 | 12 (∼0%) | 0 (0%) |

TABLE III. Gamma criteria summary for a 15 MeV beam on a phantom with layers of soft tissue, bone and lung.

### B. Execution time and efficiency gain

In this section, the absolute execution times as well as the overall speed and efficiency of GPUMCD is evaluated and compared to EGSnrc and DPM. In the following tables, $Geom_1$ is the water phantom, $Geom_2$ is the water-lung phantom and $Geom_3$ is the tissue-lung-bone phantom. $T_{EGSnrc}$ and $T_{DPM}$ are respectively the EGSnrc and DPM execution times. For the efficiency measurement, $\epsilon$, the fastest execution time of GPUMCD (with or without the optimization presented in Sec. II C 3) has been used. The acceleration factor and efficiency improvement, namely $A$ and $A_\epsilon$, also use the fastest time for GPUMCD. The acceleration factor is defined as

$$A = \frac{T_{ref}}{T_{GPUMCD}}, \tag{16}$$

and the efficiency improvement as

$$A_\epsilon = \frac{\epsilon_{GPUMCD}}{\epsilon_{ref}}. \tag{17}$$

Tab. IV presents the absolute execution times for all simulations with both electron and photon beams.

| Particles | $Geom_1$ | $Geom_2$ | $Geom_3$ |
|---|---|---|---|
| | (seconds) | | |
| Electrons | 0.118 | 0.147 | 0.162 |
| Photons | 0.269 | 0.275 | 0.366 |

TABLE IV. Absolute execution times, in seconds, for GPUMCD in the three geometries for 1M electrons and 4M photons using 15 MeV monoenergetic beams.

Tab. V presents the acceleration results for photon and electron beams where 4M photons and 1M electrons are simulated.

Tab. VI presents the results of the divergence reduction methods and acceleration strategies presented in Sec. II C 3 for electron and photon beams. In the following table $A_{comp}$, $A_{pool}$ represent the acceleration factors with respect to the base configuration execution time ($T_{base}$), with the stream compaction or *pool* divergence reduction methods enabled, respectively.

| | | Photons | | Electrons | |
|---|---|---|---|---|---|
| | | DPM | EGSnrc | DPM | EGSnrc |
| $Geom_1$ | $A$ | 453 | 1161 | 246 | 1510 |
| | $A_\epsilon$ | 515 | 1174 | 291 | 1740 |
| $Geom_2$ | $A$ | 433 | 1058 | 210 | 1264 |
| | $A_\epsilon$ | 474 | 1305 | 237 | 1403 |
| $Geom_3$ | $A$ | 203 | 947 | 225 | 1231 |
| | $A_\epsilon$ | 220 | 1050 | 242 | 1359 |

TABLE V. Acceleration factors for 1M 15 MeV electrons and 4M 15 MeV photons on the three geometries presented. Absolute execution times of 0.269 s, 0.275 s and 0.366 s were found with GPUMCD for $Geom_1$, $Geom_2$ and $Geom_3$ respectively for a photon beam and 0.118 s, 0.147 s and 0.162 s for an electron beam.

| Particles | $T_{base}$ | $A_{comp}$ | $A_{pool}$ |
|---|---|---|---|
| Electrons | 0.185 | 1.06 | 1.26 |
| Photons | 0.275 | 0.85 | 0.98 |

TABLE VI. Acceleration results of the different strategies presented in Sec. II C 3 for the second geometry ($geom_2$).

As detailed in Sec. II C 3, GPUMCD can take advantage of a system with multiple GPUs. The impact of parallelizing GPUMCD on two GPUs is presented in Tab. VII where $TPH$ (time per history) is the execution time normalized to one complete history including a primary particle and all its secondary particles, $TPH_s$ and $TPH_d$ are the execution time in single and dual GPU mode, respectively. The graphics cards used for this test are a pair of Tesla C1060. The reported execution times include every memory transfer to and from the graphics cards.

| | $TPH_s$ ($\mu$s) | $TPH_d$ ($\mu$s) | $A_{d/s}$ |
|---|---|---|---|
| Electrons | 5.43 | 2.82 | 1.92 |
| Photons | 5.20 | 2.74 | 1.90 |

TABLE VII. Execution times and acceleration factors achieved with a multi-GPU configuration.

## IV. DISCUSSION

### A. Dosimetric evaluation

Figures 1 - 3 and Tables I - III present the dosimetric evaluation performed between GPUMCD and EGSnrc. An excellent agreement is observed for homogeneous phantoms with both photon and electron beams, except for the end of the build-up region where small differences of less than 2% can be observed. In both cases, the rest of the curve as well as the overall range of the particles are not affected.

In heterogeneous simulations with low density materials, the agreement is also excellent. Differences of up to 2% can be seen in the build-up region for an incoming electron beam. These differences again do not affect the range of the particles. For a photon beam impinging on the water-lung phantom, the dose is in good agreement within as well as near the heterogeneity.

In heterogeneous simulations with a high-Z material, differences of up to 2% are found with an impinging electron beam and no differences above 1% are found for the photon beam in the presented PDD while differences of up to 1.5% are found in the profiles.

The overall quality of the dose comparison can be evaluated with the gamma value results presented in Tab. I to III. Results suggest that the code is suitable for clinical applications as the dose accuracy is within the 2%/2 mm criteria, for all cases but one, which is below the recommended calculation accuracy proposed by Van Dyk [47]. For electron beams, the gamma criteria has been well respected in the first two geometries where at most 2% of the significant voxels failed. More pronounced differences have been found in the last geometry including a high-Z material where 8% of the significant voxels fail the gamma test. However, only 3% of all significant voxels fail that test with a value higher than 1.2, indicating that

the critera are slightly too strict. Indeed, with a 2.5%-2.5mm, 3% of all significant voxels fail the test. For photon beams, all geometries meet the gamma criteria for 98 % or more of all significant voxels.

### B. Execution times

The differences in calculation efficiency between GPUMCD and the other two codes are due to three distinctive reasons: 1) differences in hardware computational power, 2) differences in the programming model and its adaptation to hardware and 3) differences in the physics simulated and approximations made. As an operation-wise equivalent CPU implementation of GPUMCD does not exist, these factors cannot be evaluated individually.

Also, the simulation setup for the test cases uses monoenergetic beams, which can favor GPU implementations. A naive GPU implementation of polyenergetic beams would lead to additional stream divergence. However, simple measures such as grouping particles with similar energy values would reduce this divergence. Future work will explore this issue in the context of clinical calculations done with polyenergetic beams.

Tab. IV shows the absolute execution times for electron and photon beams with GPUMCD. Execution times of less than 0.2 s are found for all electron cases. The execution time is higher in the simulation with lung when compared to the simulation in water likely due to the fact that more interactions are necessary. The execution time is higher in the simulation with bone because a larger number of fictitious interactions will be encountered. Similar conclusions can be taken in the cases with photon beams in which the first two simulations require less than 0.3 s and the simulation with a bone slab required more than 0.36 s.

Tab. V shows the acceleration results comparing GPUMCD to DPM and EGSnrc. It can be seen that GPUMCD is consistently faster than DPM, by a factor of at least 203x for photon beams and 210x for electron beams. The disadvantage of using the Woodcock raytracing algorithm is apparent in Tab V where the geometry featuring a heavier than water slab has the lowest acceleration factor. An acceleration of more 1200x is observed when comparing to EGSnrc for both electron beans and more than 940x for photon beams.

DPM and EGSnrc both employ much more sophisticated electron step algorithms compared to GPUMCD. Improvements to the electron-step algorithm in GPUMCD will most likely yield greater accelerations, as photon transport was shown to be much faster than the

reported coupled transport results [48], as well as improved dosimetric results.

Tab. VI presents the execution times and acceleration factors for the different divergence reduction methods and optimizations presented in Sec. II C 3. Gains of up to 26% for electron beams are observed. However, acceleration factors below 1 are found for photon beams, showing that electron beams are more divergent. These results show that the GPU architecture is not ideal for Monte Carlo simulations but that the hardware is able to cope well with this divergence. The mechanisms employed in GPUMCD to reduce this penalty have either had a negative impact on the execution time or a small positive impact that is perhaps not worth the loss in code readability and maintainability.

Tab. VII details the gain that can be obtained by executing GPUMCD on multiple graphics card. GPUMCD presents a linear growth with respect to the number of GPUs. This is expected as there is a relatively small amount of data transferred to the GPUs compared to the amount of computations required in a typical simulation. As the number of histories is reduced, so is the acceleration factor.

Efficiency improvements can be achieved with variance reduction techniques (VRT). Several methods have been published to improve the calculation efficiency of EGSnrc for instance [2, 49]. A fair comparison of GPUMCD with other Monte Carlo packages would ideally involve an optimal usage of the codes. In this paper, no VRT was used for the comparisons. It is not clear yet how VRT could potentially be implemented in GPUMCD. In the eventuality that VRT are equally applicable to each code architecture, the efficiency improvements reported in this paper would remain roughly the same. Otherwise, GPUMCD might suffer from a performance penalty compared to VRT-enabled codes.

## V. CONCLUSION AND FUTURE WORK

In this paper, a new fully coupled GPU-oriented Monte Carlo dose calculation platform was introduced. The accuracy of the code was evaluated with various geometries and compared to EGSnrc, a thoroughly validated Monte Carlo package. The overall speed of the platform was compared to DPM, an established fast Monte Carlo simulation package for dose calculations. For a 2%-2mm gamma criteria, the dose comparison is in agreement for 98% or more of all significant voxels, except in one case: the tissue-bone-lung phantom with an electron beam where 92% of significant voxels pass the gamma criteria. These results

suggest that GPUMCD is suitable for clinical use.

The execution speed achieved by GPUMCD, at least two orders of magnitude faster than DPM, let envision the use of accurate of Monte Carlo dose calculations for numerically intense applications such as IMRT or arc therapy optimizations. A 15 MeV electron beam dose calculation in water can be performed in less than 0.12 s for 1M histories while a photon beam calculation takes less than 0.27 s for 4M histories.

GPUMCD is currently under active development. Future work will include the ability to work with phase space and spectrum files as well as the integration and validation of CT-based phantoms for dose calculation. Research towards variance reduction techniques that are compatible with the GPU architecture is also planned.

## ACKNOWLEDGMENT

This work was supported by Natural Sciences and Engineering Research Council of Canada (NSERC). Dual GPU computations were performed on computers of the Réseau québécois de calcul de haute performance (RQCHP). Multiple scattering data has been provided by the National Research Council of Canada (NRC).

## REFERENCES

[1] W.R. Nelson, H. Hirayama, and D.W.O. Rogers. The EGS4 code system. *SLAC-265, Stanford Linear Accelerator Center*, 1985.

[2] I. Kawrakow. Accurate condensed history Monte Carlo simulation of electron transport. I. EGSnrc, the new EGS4 version. *Medical Physics*, 27(3):485–498, 2000.

[3] I. Kawrakow and DWO Rogers. PIRS-701: The EGSnrc Code System: Monte Carlo Simulation of Electron and Photon Transport. *Ionizing Radiation Standards (NRC, Ottawa, Ontario, 2003)*, 2003.

[4] H. Hirayama, Y. Namito, W.R. Nelson, A.F. Bielajew, and S.J. Wilderman. *The EGS5 code system*. United States. Dept. of Energy, 2005.

[5] F. Salvat, J.M. Fernández-Varea, and J. Sempau. PENELOPE-2006: a code system for Monte Carlo simulation of electron and photon transport. In *OECD/NEA Data Bank*, 2006. Available at `http://www.nea.fr/science/pubs/2006/nea6222-penelope.pdf`.

[6] J. Baro, J. Sempau, JM Fernandez-Varea, and F. Salvat. PENELOPE: an algorithm for Monte Carlo simulation of the penetration and energy loss of electrons and positrons in matter. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, 100(1):31–46, 1995.

[7] DWO Rogers. Fifty years of Monte Carlo simulations for medical physics. *Physics in medicine and biology*, 51:R287, 2006.

[8] R. Mohan, C. Chui, and L. Lidofsky. Differential pencil beam dose computation model for photons. *Medical Physics*, 13(1):64–73, 1986.

[9] Anders Ahnesjö, Mikael Saxner, and Avo Trepp. A pencil beam model for photon dose calculation. *Medical Physics*, 19(2):263–273, 1992.

[10] U Jelen, M Söhn, and M Alber. A finite size pencil beam for IMRT dose optimization. *Physics in Medicine and Biology*, 50(8):1747, 2005.

[11] T. R. Mackie, J. W. Scrimger, and J. J. Battista. A convolution method of calculating dose for 15-MV x rays. *Medical Physics*, 12(2):188–196, 1985.

[12] H. Helen Liu, T. Rock Mackie, and Edwin C. McCullough. A dual source photon beam model used in convolution/superposition dose calculations for clinical megavoltage x-ray beams. *Medical Physics*, 24(12):1960–1974, 1997.

[13] T. Krieger and O.A. Sauer. Monte Carlo-versus pencil-beam-/collapsed-cone-dose calculation in a heterogeneous multi-layer phantom. *Physics in medicine and biology*, 50:859, 2005.

[14] George X. Ding, Joanna E. Cygler, Christine W. Yu, Nina I. Kalach, and G. Daskalov. A comparison of electron beam dose calculation accuracy between treatment planning systems using either a pencil beam or a Monte Carlo algorithm. *International Journal of Radiation Oncology Biology Physics*, 63(2):622 – 633, 2005.

[15] C-M Ma, T Pawlicki, S B Jiang, J S Li, J Deng, E Mok, A Kapur, L Xing, L Ma, and A L Boyer. Monte Carlo verification of IMRT dose distributions from a commercial treatment planning optimization system. *Physics in Medicine and Biology*, 45(9):2483, 2000.

[16] Iwan Kawrakow, Matthias Fippel, and Klaus Friedrich. 3D electron dose calculation using a Voxel based Monte Carlo algorithm (VMC). *Medical Physics*, 23(4):445–457, 1996.

[17] Matthias Fippel. Fast Monte Carlo dose calculation for photon beams based on the VMC electron algorithm. *Medical Physics*, 26(8):1466–1475, 1999.

[18] J. Gardner, J. Siebers, and I. Kawrakow. Dose calculation validation of VMC++ for photon beams. *Medical Physics*, 34(5):1809–1818, 2007.

[19] Josep Sempau, Scott J Wilderman, and Alex F Bielajew. DPM, a fast, accurate Monte Carlo code optimized for photon and electron radiotherapy treatment planning dose calculations. *Physics in Medicine and Biology*, 45(8):2263–2291, 2000.

[20] O. Kutter, R. Shams, and N. Navab. Visualization and GPU-accelerated simulation of medical ultrasound from CT images. *Computer methods and programs in biomedicine*, 94(3):250–266, 2009.

[21] M. de Greef, J. Crezee, J. C. van Eijk, R. Pool, and A. Bel. Accelerated ray tracing for radiotherapy dose calculations on a GPU. *Medical Physics*, 36(9):4095–4102, 2009.

[22] Sami Hissoiny, Benoît Ozell, and Philippe Després. A convolution-superposition dose calculation engine for GPUs. *Medical Physics*, 37(3):1029–1037, 2010.

[23] X. Jia, Y. Lou, R. Li, W.Y. Song, and S.B. Jiang. GPU-based fast cone beam CT reconstruction from undersampled and noisy projection data via total variation. *Medical physics*, 37:1757, 2010.

[24] Vasily Volkov and James W. Demmel. Benchmarking GPUs to tune dense linear algebra. In *SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, pages 1–11, Piscataway, NJ, USA, 2008. IEEE Press.

[25] J.A. Anderson, C.D. Lorenz, and A. Travesset. General purpose molecular dynamics simulations fully implemented on graphics processing units. *Journal of Computational Physics*, 227(10):5342–5359, 2008.

[26] S. Chen, J. Qin, Y. Xie, J. Zhao, and P.A. Heng. A fast and flexible sorting algorithm with cuda. *Algorithms and Architectures for Parallel Processing*, pages 281–290, 2009.

[27] T. Preis, P. Virnau, W. Paul, and J.J. Schneider. GPU accelerated Monte Carlo simulation of the 2D and 3D Ising model. *Journal of Computational Physics*, 228(12):4468–4477, 2009.

[28] M. Januszewski and M. Kostur. Accelerating numerical solution of stochastic differential equations with CUDA. *Computer Physics Communications*, 181(1):183–188, 2010.

[29] Erik Alerstam, Tomas Svensson, and Stefan Andersson-Engels. Parallel computing with graphics processing units for high-speed Monte Carlo simulation of photon migration. *Journal of*

*Biomedical Optics*, 13(6):060504, 2008.

[30] Andreu Badal and Aldo Badano. Accelerating Monte Carlo simulations of photon transport in a voxelized geometry using a massively parallel graphics processing unit. *Medical Physics*, 36(11):4878–4880, 2009.

[31] Xun Jia, Xuejun Gu, Josep Sempau, Dongju Choi, Amitava Majumdar, and Steve B Jiang. Development of a GPU-based Monte Carlo dose calculation code for coupled electron-photon transport. *Physics in Medicine and Biology*, 55(11):3077, 2010.

[32] L. L. Carter, E. D. Cashwell, and W. M. Taylor. Monte Carlo Sampling with Continuously Varying Cross Sections Along Flight Paths. *Nucl. Sci. Eng.*, 48:403–411, 1972.

[33] O. Klein and T. Nishina. Uber die Streuung von Strahlung durch freie Elektronen nach der neuen relativistischen Quantendynamik von Dirac. *Zeitschrift fur Physik A Hadrons and Nuclei*, 52(11):853–868, 1929.

[34] CJ Everett, ED Cashwell, and GD Turner. A new method of sampling the Klein-Nishina probability distribution for all incident photon energies above 1 keV. Technical report, LA–4663, Los Alamos Scientific Lab., N. Mex., 1971.

[35] Harold Elford Johns and John Robert Cunningham. *The Physics of Radiology, fourth edition*, page 181. Charles C Thomas, Springfield IL, 1983.

[36] F. Sauter. Atomic photoelectric effect in the K-shell according to the relativistic wave mechanisms of Dirac. *Ann. Phys.(Leipzig)*, 11:454–488, 1931.

[37] I. Kawrakow and DWO Rogers. The EGSnrc code system: Monte Carlo simulation of electron and photon transport, NRCC Report PIRS-701. *Ottawa (ON): National Research Council of Canada*, 2006.

[38] M.J. Berger. Monte Carlo calculation of the penetration and diffusion of fast charged particles. *Methods in computational physics*, 1:135–215, 1963.

[39] Iwan Kawrakow and Alex F. Bielajew. On the representation of electron multiple elastic-scattering distributions for Monte Carlo calculations. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, 134(3-4):325 – 336, 1998.

[40] Sami Hissoiny, Benoît Ozell, and Philippe Després. Fast convolution-superposition dose calculation on graphics hardware. *Medical Physics*, 36(6):1998–2005, 2009.

[41] George Marsaglia and Arif Zaman. A New Class of Random Number Generators. *The Annals of Applied Probability*, 1(3):462.480, 1991.

[42] Pierre L'ecuyer and Richard Simard. TestU01: A C library for empirical testing of random number generators. *ACM Trans. Math. Softw.*, 33(4):22, 2007.

[43] M. Billeter, O. Olsson, and U. Assarsson. Efficient stream compaction on wide SIMD many-core architectures. In *Proceedings of the 1st ACM conference on High Performance Graphics*, pages 159–166. ACM, 2009.

[44] Timo Aila and Samuli Laine. Understanding the Efficiency of Ray Traversal on GPUs. In *Proc. High-Performance Graphics*, 2009.

[45] I. Kawrakow and M. Fippel. Investigation of variance reduction techniques for Monte Carlo photon dose calculation using XVMC. *Physics in medicine and biology*, 45:2163, 2000.

[46] Daniel A. Low and James F. Dempsey. Evaluation of the gamma dose distribution comparison method. *Medical Physics*, 30(9):2455–2464, 2003.

[47] J. Van Dyk et al. *The modern technology of radiation oncology*, pages 252–253. Medical Physics Publ., 1999.

[48] Sami Hissoiny, Benoît Ozell, and Philippe Després. GPUMCD, a new GPU-oriented Monte Carlo dose calculation platform. In *XVIth ICCR Conference Proceedings*, 31 May - 3 June 2010. Amsterdam, NL.

[49] J. Wulff, K. Zink, and I. Kawrakow. Efficiency improvements for ion chamber calculations in high energy photon beams. *Medical physics*, 35:1328, 2008.